Advanced Technical Skills (ATS) North America

Running MPI applications on Linux over Infiniband cluster with Intel MPI

IBM High Performance Computing
February 2010

Y. Joanna Wong
yjw@us.ibm.com

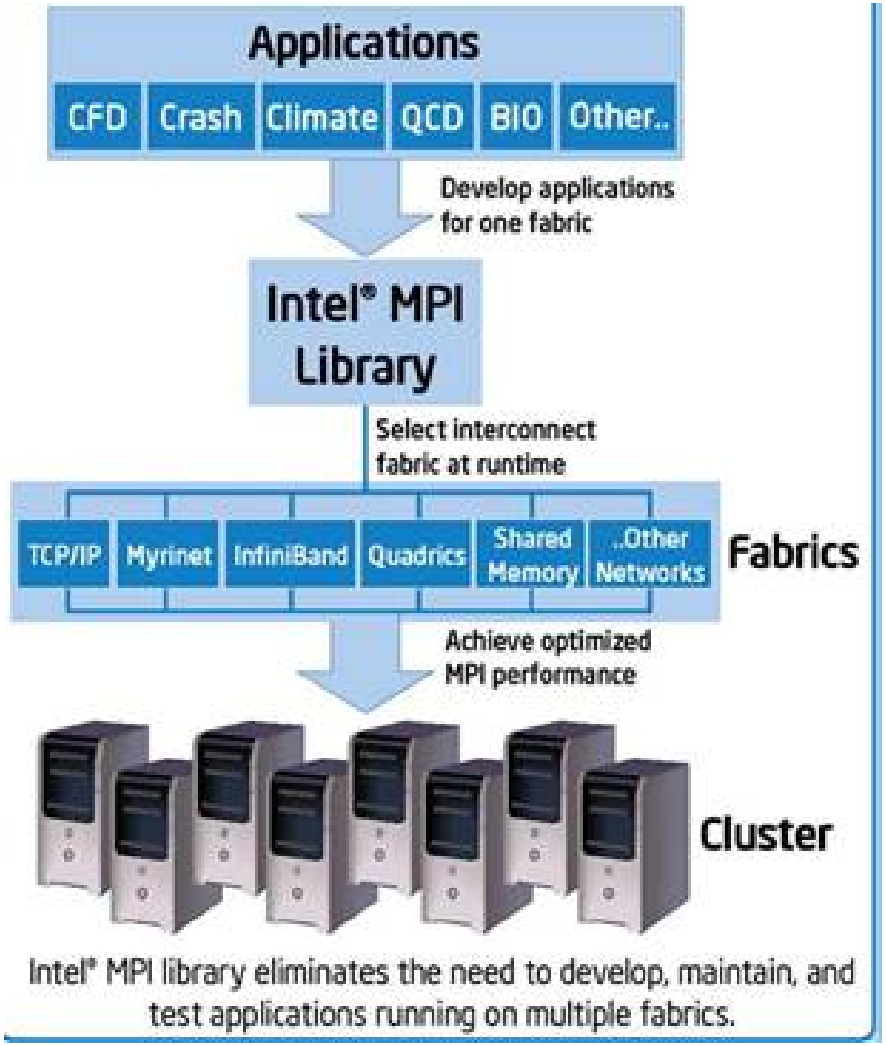
Intel MPI Library

- **Multi-fabric message passing library based on**
  - MPICH2 implementation of Message Passing Interface v2 (MPI-2) from Argonne National Lab
  - In part on InfiniBand Architecture RDMA drivers from MVAPICH2 from Ohio State University Network-Based Computing Laboratory
- **Switch interconnection fabrics support without re-linking**
  - Sock
    - TCP/IP sockets – ethernet, IPoIB
  - SHM – shared memory for large SMPs
  - SSM – shared memory + sockets
  - RDMA – support for RDMA enabled fabrics
    - InfiniBand, Myrinet, Quadrix
    - Implemented using DPAL
  - RDSSM (RDMA + SHM + Sock)
    - Shared memory for intra-node processes
    - RDMA for inter-node processes
    - Fails over to sockets if RDMA device is not available (default)

Intel MPI…

- **The latest version is Intel MPI Library 3.2**
  - Faster startup and collective operation algorithms with improved performance
  - Greater scalability over sockets and shared memory
  - Added support on Linux for Intel Compiler 11.0 and DAPL 2.0 (in addition to 1.1 and 1.2 DAPL compliant environment)
  - Tested interoperability with GNU compilers 3.3 and higher
  - Easily integrated with several Linux Job schedulers including Torque 1.2.0 and higher
  - Distribution:
    - Free runtime environment for pre-installation or redistribution
    - SDK includes compilation tools, interface (static) libraries, debug libraries, trace libraries, include files and modules, and test codes
    - Distributed also with Intel Cluster Toolkit 3.2
- **Documentation**
  - Intel MPI Library Getting Started Guide
  - Intel MPI Library Reference Manual

Intel MPI Library



Compiling MPI applications

- Compiler commands are wrapper scripts that will generate the correct flags, compiler options, includes, defines and libraries to add to the compile and link commands
- For compiler commands mpicc, mpicxx, mpif77, mpif90, the underlying compilers are the GNU compilers: C, C++, Fortran77 3.3 or higher, Fortran 95 4.0 or higher
- For compiler commands mpiicc, mpiicpc, mpiifort, the underlying compilers are the Intel C, C++ and Fortran compilers version 9.x, 10.x, 11.x
- Can override underlying compilers with environment variables or command line option:

	MPI compiler	Environment variable	Command line option
C	mpicc	MPICH_CC or I_MPI_CC	-cc=<compiler>
C++	mpicxx	MPICH_CXX or I_MPI_CXX	-cxx=<compiler>
F77	mpif77	MPICH_F77 or I_MPI_F77	-f77=<compiler> or -fc=<compiler>
F90	mpif90	MPICH_F90 or I_MPI_F90	-f90=<compiler> or -fc=<compiler>

- Use the MPI compiler option -show to display the compile and link commands

  /opt/intel/imp/3.2/bin64/mpicc –show –c test.c

  shows: gcc -c test.c –I/opt/intel/impi/3.2/include64

  /opt/intel/impi/3.2/bin64/mpiicc –show –c test.c

  shows: icc -c test.c -I/opt/intel/impi/3.2/include64

- Building MPI applications with Intel MPI installed under directory $I_MPI_ROOT:

	Intel 64 architecture	IA-32 architecture
Binaries, script and executables	$I_MPI_ROOT/bin64	$_IMPI_ROOT/bin
Libraries and compiler input files	$I_MPI_ROOT/lib64	I_IMPI_ROOT/lib
Include and header files	$I_MPI_ROOT/include 64	$I_MPI_ROOT/include
Additional configuration files	$I_MPI_ROOT/etc64	$I_MPI_ROOT/etc

Running MPI applications with Intel MPI Library

- **Compile and link the application with compiler commands**

  $ source /opt/intel/Compiler/11.0/069/bin//iccvars.sh intel64

  $ source /opt/intel/impi/3.2/bin64/mpivars.sh

  $ mpiicc –O –o test.icc test.c

- Set up .mpd.conf file
  - Create the file $HOME/.mpd.conf with one line

    secretword=*mpd_secret_word*
  - $ chmod 600 $HOME/.mpd.conf
- Set up mpd.hosts file
  - Create mpd.hosts file with one line per host e.g node001 and node002

    node01

    node002
  - make sure can ssh to the hosts in mpd.hosts without being prompted for password.

Running MPI applications with Intel MPI Library

- Start mpd daemons with mpdboot
  - $ mpdboot –n <# nodes> -r /usr/bin/ssh –f mpd.hosts
- Use mpdtrace -l to show full hostnames, listening port, and interface of mpds

  $ mpdtrace –l

  node001_36396 (172.20.101.1)

  node002_33512 (172.20.101.2)
- To start mpd daemons on IPoIB interfaces
  - Create mpd.hosts file with one line per host with IB interfaces e.g node001 and node002

    node001 ifhn=node001-ib0

    node002 ifhn=node002-ib0
  - Start mpd daemons on compute server node001

    $ mpdboot –r ssh –n 2 –f mpd.hosts –ifhn=node001-ib0

    $ mpdtrace –l

    node001_36383 (192.168.101.1)

    node002_33509 (192.168.101.2)

Running MPI applications with Intel MPI Library

- Select network fabric, e.g. RDMA + share memory

  $ export I_MPI_DEVICE=rdssm

- Create machine file <machine_file> with one host name per line

  node001 ifhn=node001-ib0
  node001 ifhn=node001-ib0
  node002 ifhn=node002-ib0
  node002 ifhn=node002-ib0

  or

  node001:2 ifhn=node001-ib0
  node002:2 ifhn=node002-ib0

- Run MPI program with mpiexec

  $ mpiexec –machinefile <machine_file> –envall –np <# processes> <executable>

Running MPI applications with Intel MPI Library

- Syntax of mpiexec

  mpiexec <global-options> <local-options> <MPI executable>

  – Commonly used global-options

    -nolocal
    -perhost <# of processes>
    -machinefile <machine_file>
    -genv <ENVAR> <value>
    -ifhn <hostname>

  – Commonly used local-options

    -n or –np <# of processes>
    -env <ENVAR> <value>
    -envall

- mpirun – simplified job startup command

  mpirun [mpdboot options] [mpiexec options]

  – The first non-mpdboot option (including –n or –np) delimits mpdboot options

Running MPI applications with Intel MPI Library

- Intel MPI Library environment variables
  - I_MPI_DEVICE select particular network fabric to be used
    - rdma RDMA-capable including InfiniBand
    - rdssm combined TCP + shared memory + rdma
  - I_MPI_FALLBACK_DEVICE
    - valid for rdma and rdssm nodes
    - {enable, yes, on, 1} fall back upon ssm (TCP +shared memory) if initialization of DAPL fabric fails
    - {disable, no, off, 0} terminate job if selected fabric cannot be initialized
  - I_MPI_DEBUG
    - To positively confirm I_MPI_DEVICE use, set variable to 2 or higher

Running MPI applications with Intel MPI Library

I_MPI_DEBUG=2

I_MPI_DEVICE=rdssm

I_MPI_FALLBACK_DEVICE=disable

Successful:

I_MPI: [0] MPIDI_CH3I_RDMA_init(): will use DAPL provider from registry: OpenIB-cma

I_MPI: [0] MPIDI_CH3_Init(): will use rdssm configuration

I_MPI: [0] LIBRARY pinning(): The process is pinned on hpc001:CPU00

I_MPI: [0] MPI_Init: The process (pid=17300) started on hpc001

Failed:

I_MPI: [1] MPIDI_CH3I_RDMA_init(): will use DAPL provider from registry: OpenIB-cma

I_MPI: [1] MPIDI_CH3I_RDMA_check_dapl_provider_mismatch(): DAPL provider on ra 0:hpc003 OpenIB-cma1.2 != on rank 1:hpc004

[1] DAPL provider is not found and fallback device is not enabled

[cli_3]: aborting job:

Running MPI applications with Intel MPI Library

- I_MPI_DEVICE=rdma failed with message

  libibverbs: Warning: RLIMIT_MEMLOCK is 32768 bytes.

  This will severely limit memory registrations

- With OFED, default max locked memory limit in Linux kernel is usually low for HPC applications

  max locked memory (kbytes, -l) 32

- Need to set available locked memory to a larger number (e.g. unlimited). Check with ulimit –l or ulimit –a

  max locked memory (kbytes, -l) unlimited

I_MPI_DEVICE=rdma

- Default DAPL library is first entry in file /etc/dat.conf for the Infiniband device
  - /usr/sbin/ibstatus will list name of Infiniband device
    - Infiniband device 'mlx4_0' port 1 status:
    - default gid: fe80:0000:0000:0000:0002:c903:0005:9189
    - base lid: 0xd5
    - sm lid: 0x1
    - state: 4: ACTIVE
    - phys state: 5: <unknown>
    - rate: 20 Gb/sec (4X DDR)
- To use alternative RDMA providers defined in /etc/dat.conf, used I_MPI_DEVICE=rdma:*provider*

Example 1: DAPL 1.2 and SCM

OpenIB-mlx4_0-1 u1.2 nonthreadsafe default libdaplscm.so.1 dapl.1.2 "mlx4_0 1" “”

I_MPI_DEVICE=rdma:OpenIB-mlx4_0-1

Example 2: DAPL 2.0 and SCM

ofa-v2-mlx4_0-1 u2.0 nonthreadsafe default libdaploscm.so.2 dapl.2.0 "mlx4_0 1" ""

I_MPI_DEVICE=rdma: ofa-v2-mlx4_0-1

Advanced Technical Skills (ATS) North America

Intel Math Kernel Library

IBM High Performance Computing
February 2010

Y. Joanna Wong
yjw@us.ibm.com

Intel Math Kernel Library (MKL) 10.2

- Previous MKL and MKL cluster edition merged to one package
- Extensively threaded math routines including BLAS, LAPACK, ScaLAPACK, Sparse Solvers, Fast Fourier Transform, and Vector Math
- Highly optimized for current and next-generation Intel processors
- Automatic runtime processor detection
- Included FFTW interfaces

Linking with MKL library

- Layered Model Concept in version 10.x
- 4 layers of libraries in MKL 10 and one library to link from each layer
  - Interface layer: LP64 and IPL interfaces
  - Threading layer
  - Computation layer
  - Compiler Support Run-time Libraries
- Linking

  <MPI linker script> <files to link> -L<MKL path> <MKL library> <BLACS> <MKL core libraries>
- With static link, the interface layer, threading layer, and computation layer libraries are enclosed with group symbols: -Wl,--start-group ……. -Wl,--end-group

Example: Linking with ScaLAPACK

with Intel MPI:

**mpiicc <files to link> -L<MKL path> **

-lmkl_scalapack_lp64 \

-lmkl_blacs_intelmpi_lp64

-lmkl_intel_lp64 –lmkl_intel_thread –lmkl_core -lmkl_solver_lp64 \

–liomp5 -lpthread